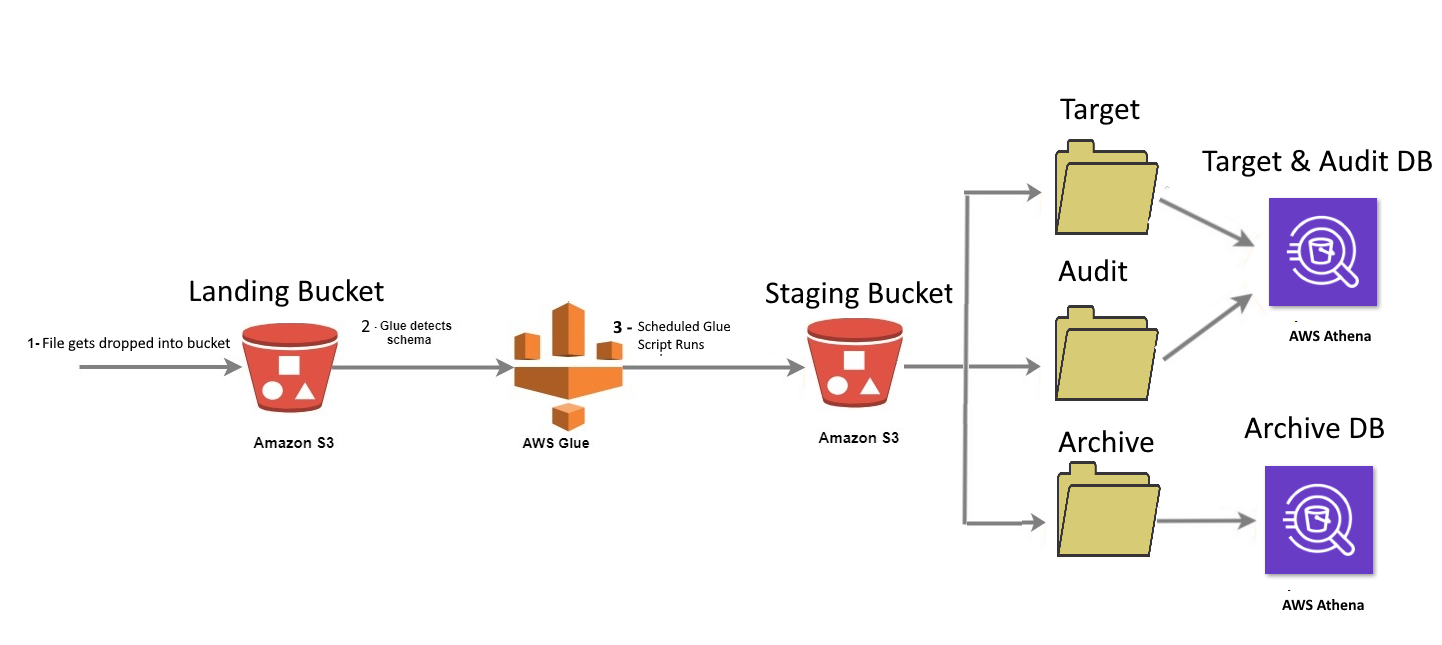
**Wavicle to CCAO ETL Code Handoff Readme**



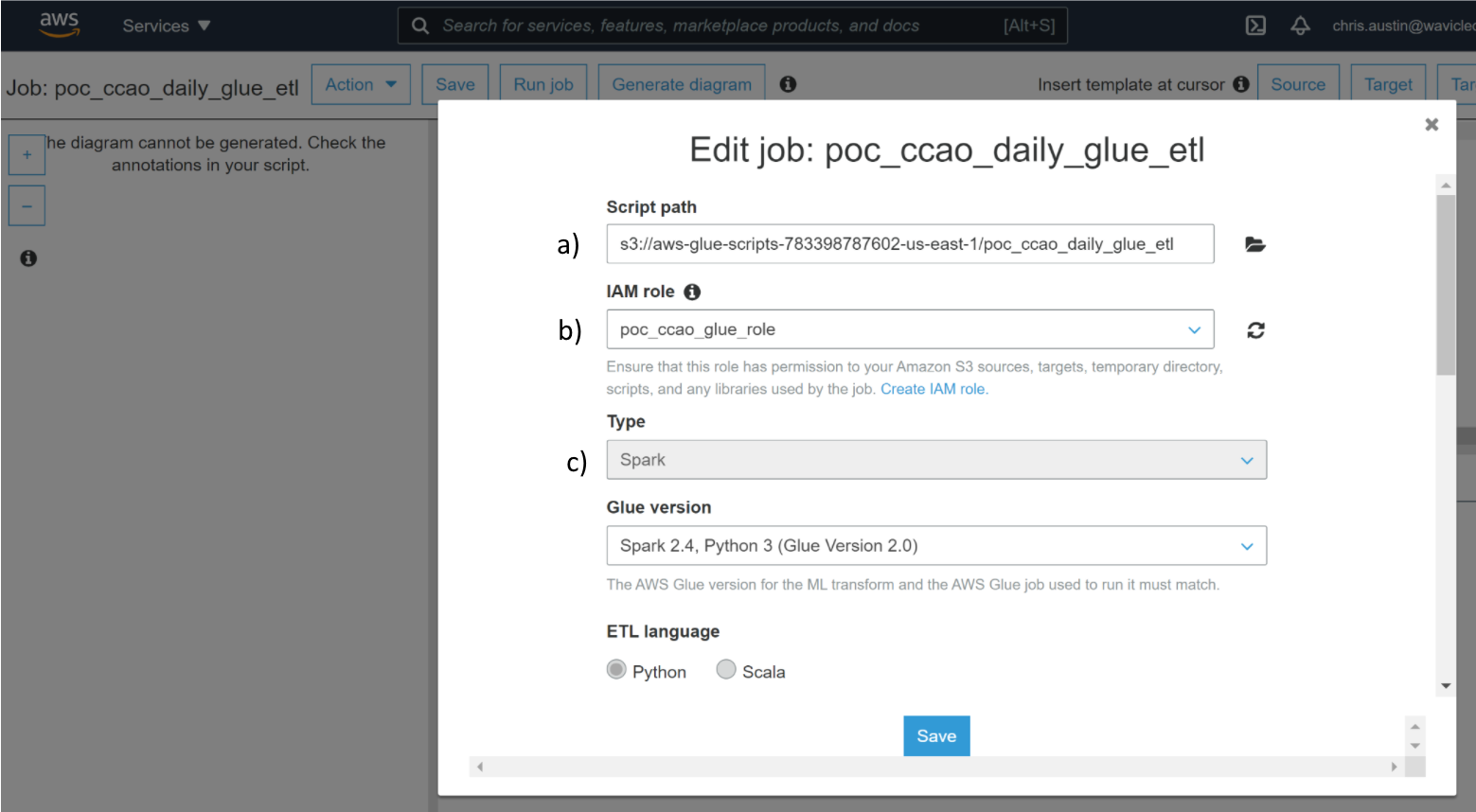
**Table of Contents**

1. Glue Job Configuration
2. YAML Configuration File
3. Glue Job Code (ETL Code)
4. Glue Crawlers
5. Athena Tables
6. Errors to Watch out For
7. Paths
8. Simple Walkthrough Example

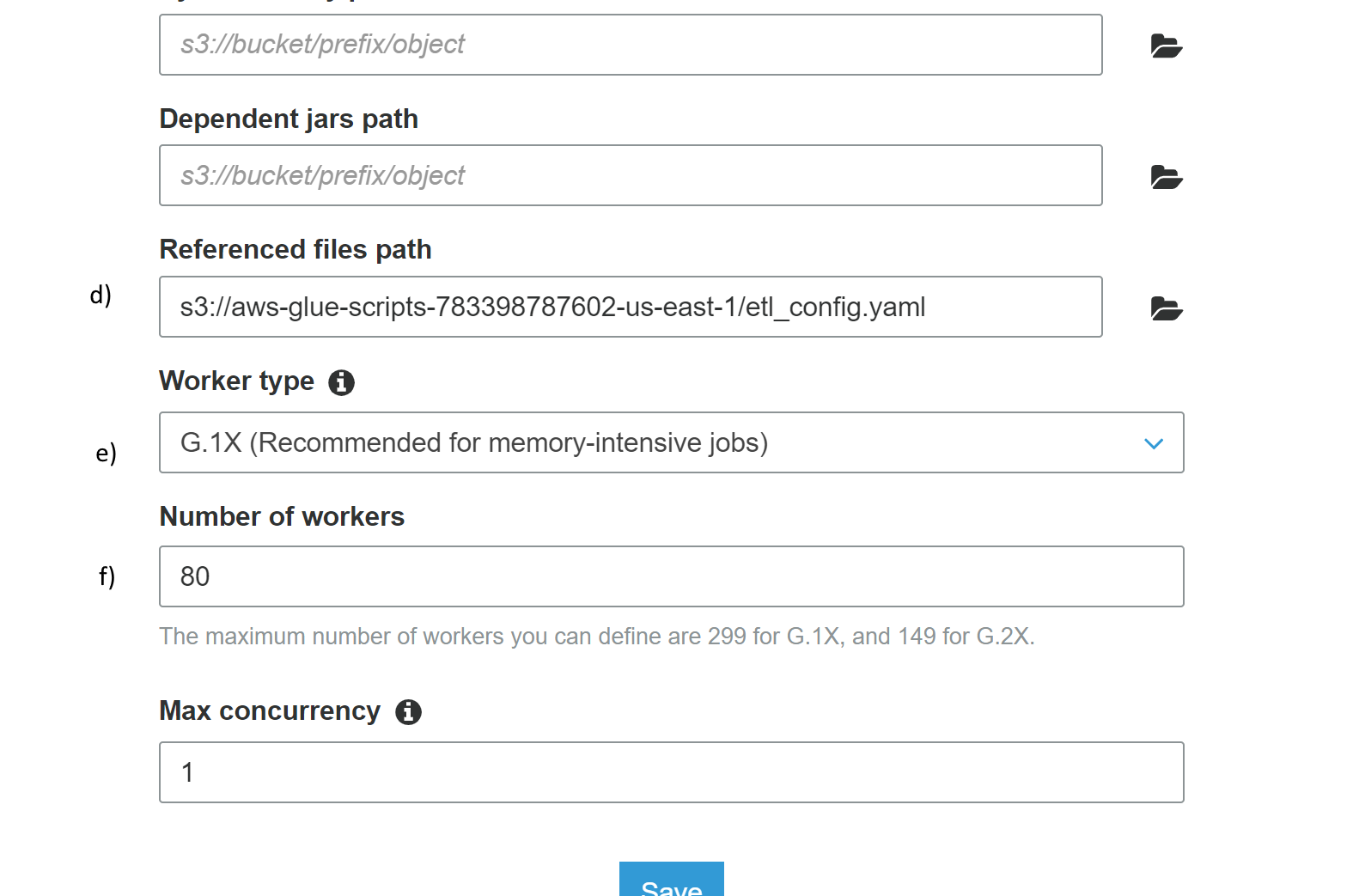
Explanation Table of Contents, each section contains a (things - to - watch out for) section

1. **Glue Job configuration**

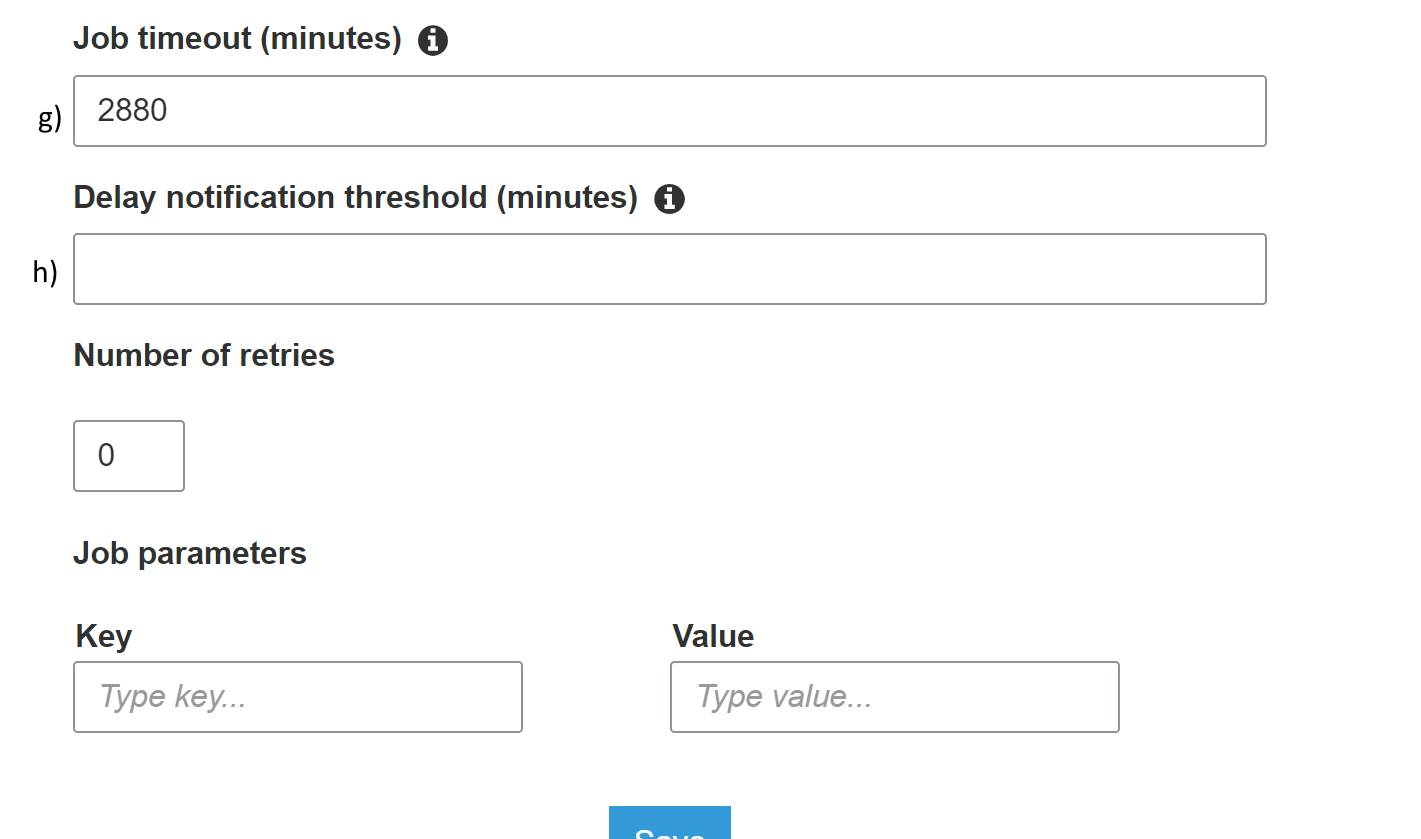
Below are explanations with pictures explaining the current configuration and what is important to know moving forward



1. Script Path (ref 7g) will output the physical copy of the script to be run by the job
2. The IAM Role must be picked to execute the job ( it should have full s3 and glue access)
3. Type and Glue Version must be kept the same, or for future updates be in alignment with what AWS Glue and Apache Spark are compatible with. See (ref 7h)



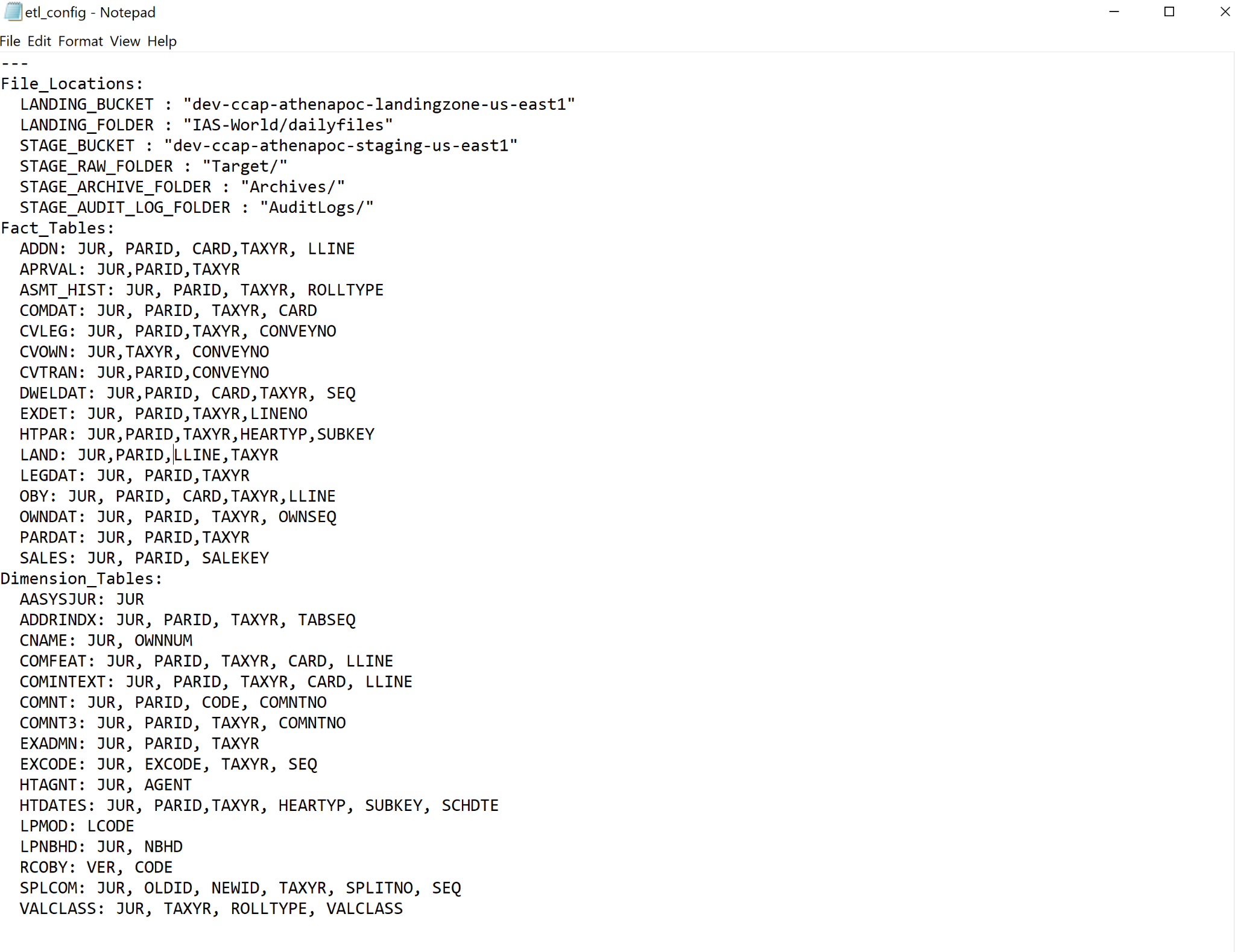
1. Referenced file path allows Glue to easily access any files you include, for now it only includes the config file
2. Worker type picks the type of works (not driver/master node) that AWS glue will distribute the work to
3. Number of workers and Max concurrency will specify the number of workers per job (max 299), max concurrency is how many instances of this job can run at the same time.



1. Job timeout simply sets an upper limit on how long the job can take before it is cancelled for taking too long
2. The other options in the Glue job configuration are delay notification, # of retries, and job parameters. All of which we are not using and are irrelevant at the time of handoff.
3. **YAML Configuration File**

The config file is written in yaml format due to it’s compatibility with python which the POC is written in.  
For the reader’s convenience if you would like a general introduction to yaml please see the link below

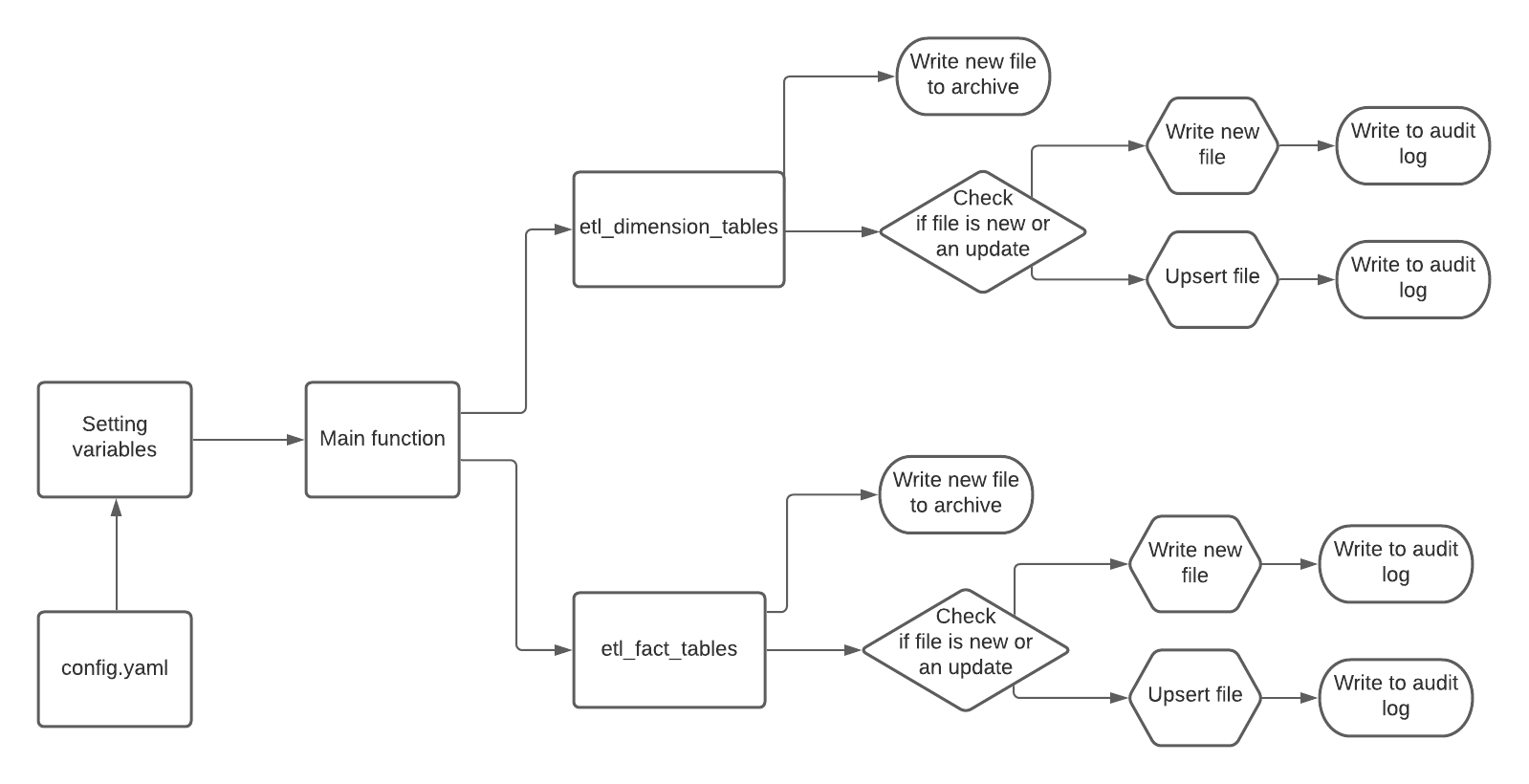
<https://www.cloudbees.com/blog/yaml-tutorial-everything-you-need-get-started>



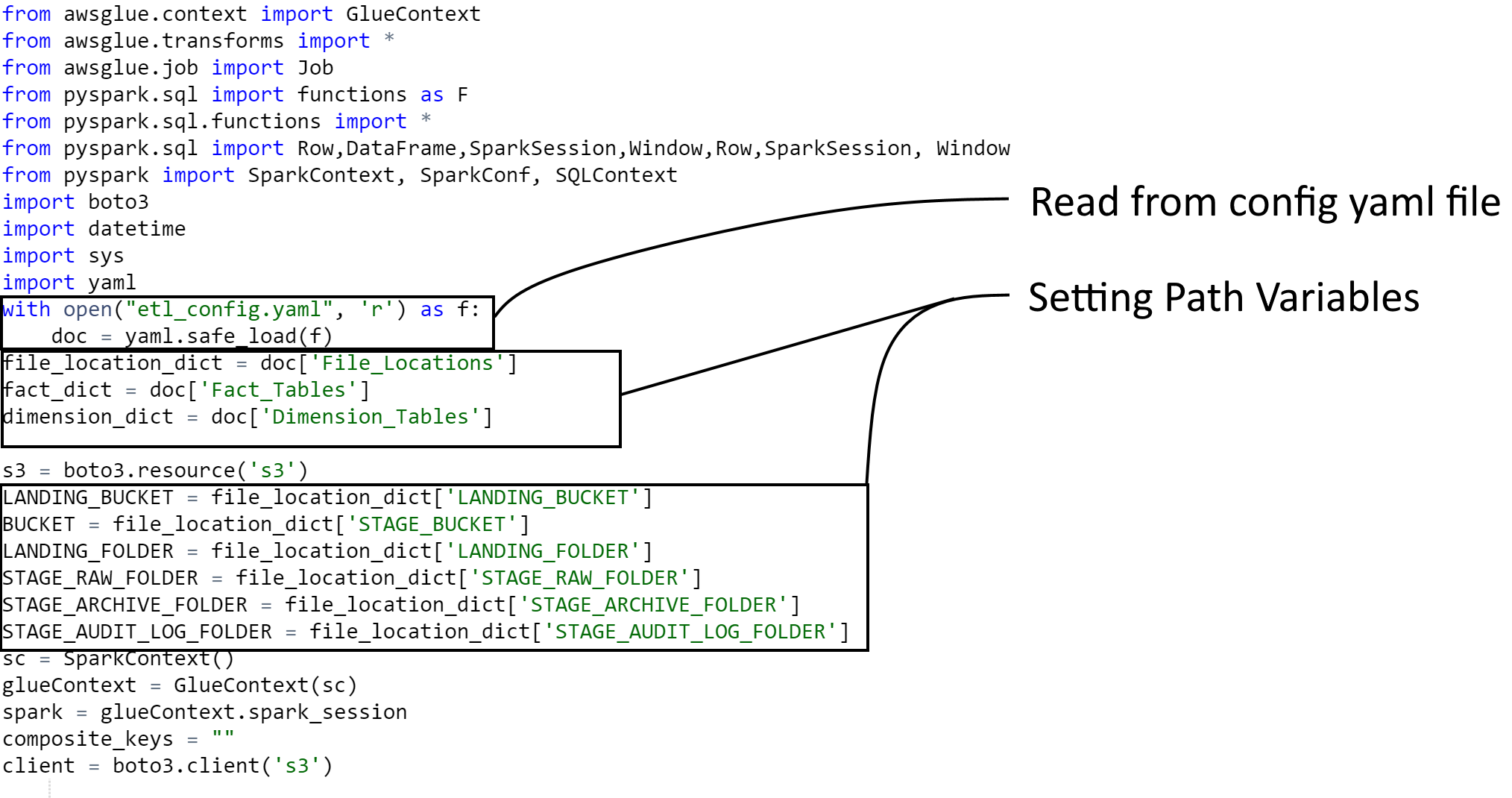
Things-to-watch out for

1. The values for all three parameters must be comma delimited for the program to parse them, the way it is setup
2. When creating a yaml file or editing it, if you copy and paste from somewhere else it sometimes creates an issue with how the characters are encoded into the file which can then result in the aws glue job not properly being able to read the file.
3. The format which the file is currently in regarding colons must be used. The program is only setup to parse three parameters (File\_location,Fact\_tables,Dimension\_tables) and the key,value pairs associated with them.
4. **Glue job code**

When the python code runs it will follow the process shown below. It is of note however that since the program uses pyspark it will not actually run completely linearly because pyspark will partition the data and automatically divy up the work for fastest processing time.

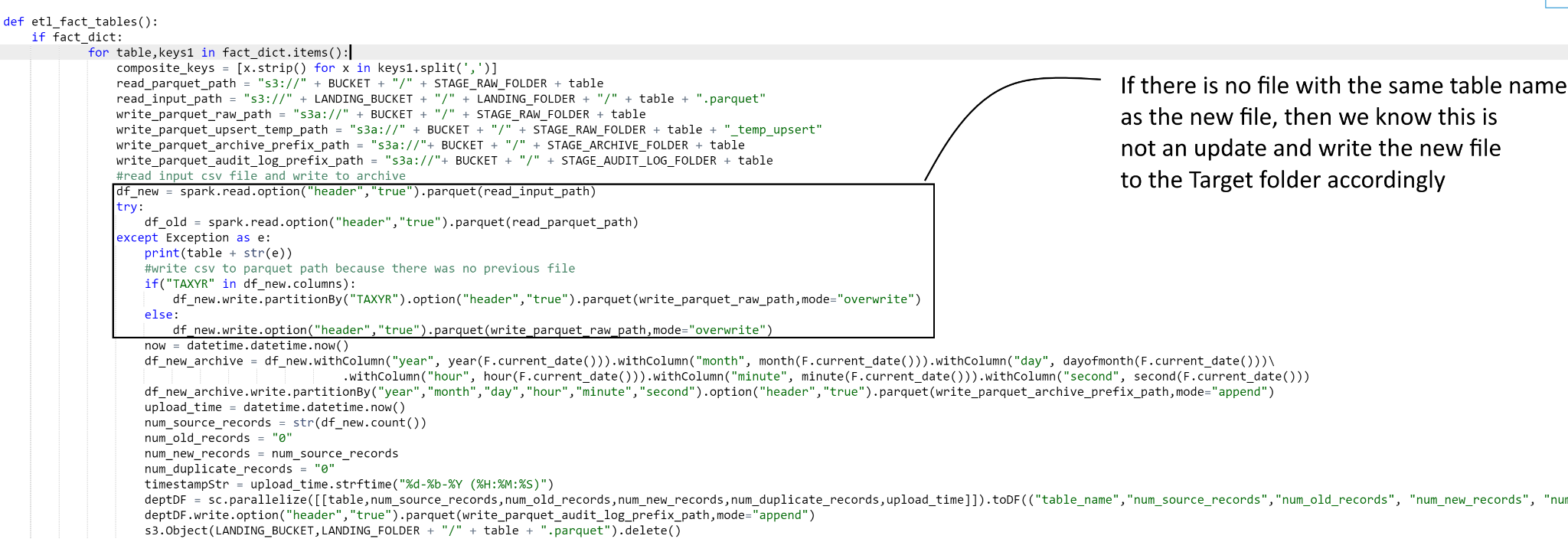


The first step in the code is to grab the etl\_config.yaml file and read from it, followed by setting the global variables needed by the program to read and write from s3.   
*See the below image to observe where that occurs in the code*



After all the global variables have been set the main function will be called which will call the function move\_from\_landing\_to\_stage() which will subsequently call the functions etl\_dimension\_tables() and etl\_fact\_tables() to begin the etl process for both.  
*See the below image to observe where that occurs in the code*  


The code will check if there is an old copy of the file using a try statement, if there is not then it will immediately begin the process of writing a new file  
*See the below image to observe where that occurs in the code*



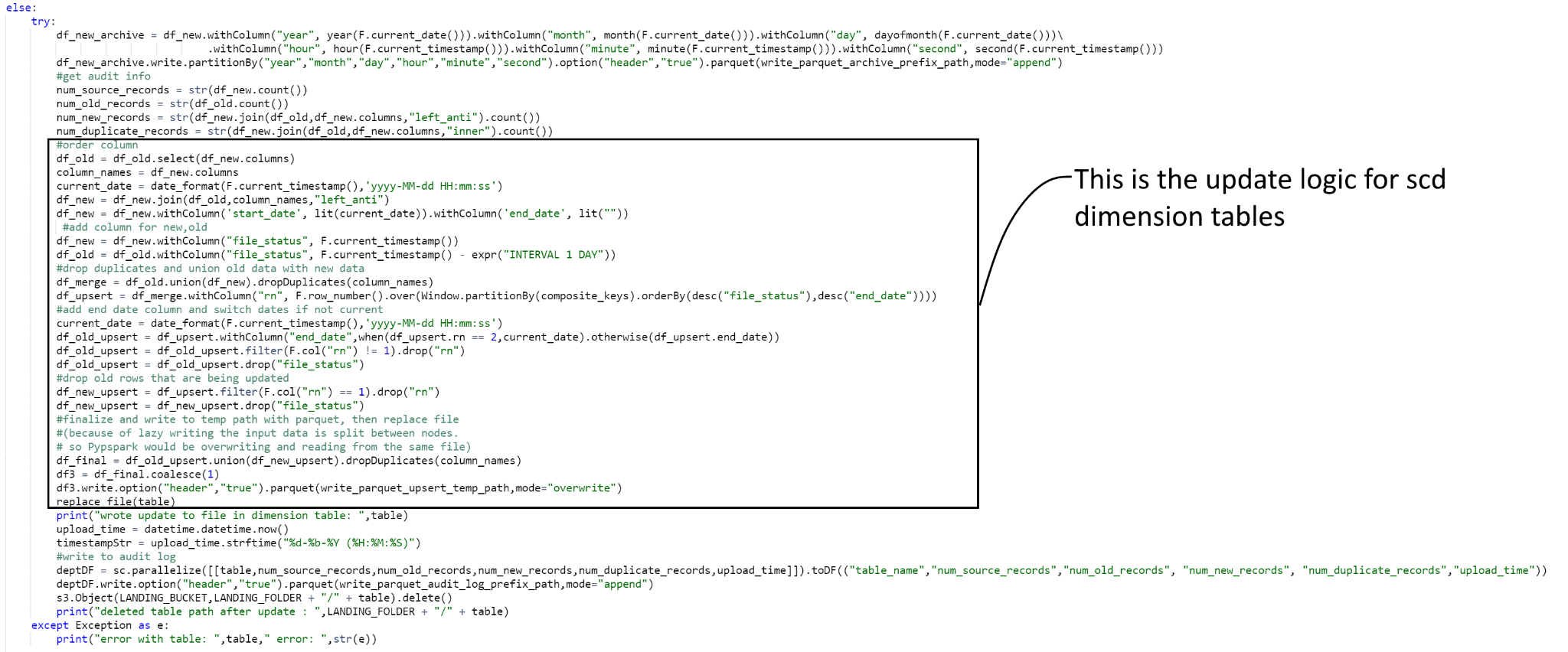
If the code did find an old copy of the file already in the Target folder it will begin the process of updating the file using the non-scd upsert logic as shown below.  
*See the below image to observe where that occurs in the code*



In the etl\_dimension\_tables() function, it will follow a very similar flow the the etl\_fact\_tables() function and begin by checking if there is an old copy of the file using a try statement, if there is not then it will immediately begin the process of writing a new file  
*See the below image to observe where that occurs in the code*



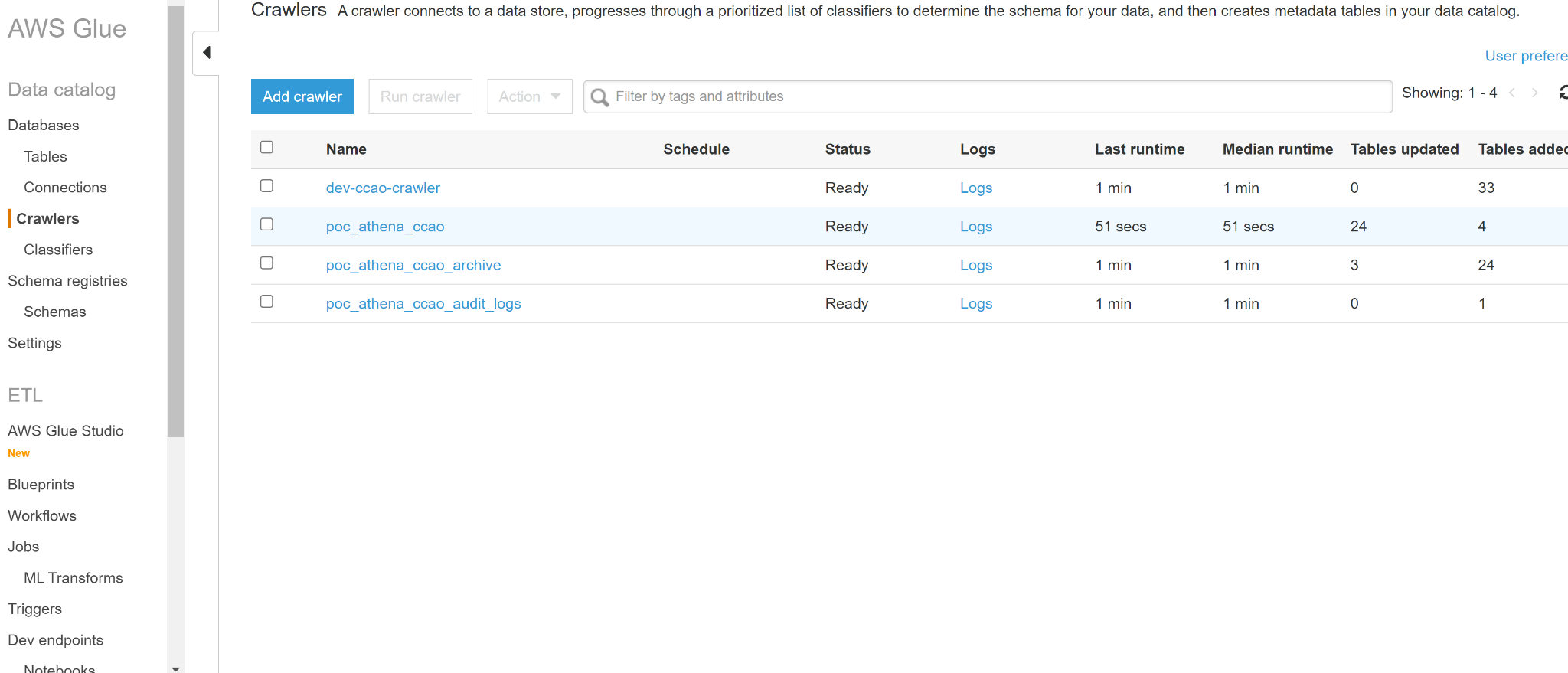
If the code did find an old copy of the file already in the Target folder it will begin the process of updating the file using the scd upsert logic as shown below.  
*See the below image to observe where that occurs in the code*



Below the update functions in both dimension and fact tables, there will be auditing and archive of the new file, as well as the file being deleted once the update, audit, and archiving are done. The version of the code in github is slightly different regarding deletion than in this file, because of the change in which the parquet file is saved from “filename.parquet” to “filename/” with the parquet files inside the filename folder. This is because the contents inside the folder need to be deleted instead of only the file associated with the file name.

1. **Glue Crawlers**

There are 3 glue crawlers shown below.



They are as follows

1. Poc\_athena\_ccao

-This crawler will setup the athena tables based on the files in the Target Folder  
s3://dev-ccap-athenapoc-staging-us-east1/Target/

1. Poc\_athena\_ccao\_archive

-This crawler will setup the athena tables based on the files in the ArchiveFolder  
s3://dev-ccap-athenapoc-staging-us-east1/Archives/

1. poc\_athena\_ccao\_audit\_logs

-This crawler will setup the athena tables based on the files in the AuditLogs folder  
s3://dev-ccap-athenapoc-staging-us-east1/AuditLogs/

1. **Athena Tables**

List of tables in dev\_poc\_ccao:

1. Aasysjur
2. addn
3. addrindx
4. aprval
5. asmt\_hist
6. auditlogs
7. cname
8. comdat
9. comfeat
10. comintext
11. comnt
12. cvleg
13. cvown
14. cvtran
15. dweldat
16. exadmn
17. excode
18. exdet
19. htagnt
20. htdates
21. htpar
22. land
23. legdat
24. lpmod
25. lpnbhd
26. oby
27. owndat
28. pardat
29. rcoby
30. sales
31. splcom
32. Valclass

List of tables in dev\_poc\_ccao:

1. archive\_aasysjur
2. archive\_addn
3. archive\_addrindx
4. archive\_aprval
5. archive\_asmt\_hist
6. archive\_cname
7. archive\_comdat
8. archive\_comnt
9. archive\_cvleg
10. archive\_cvown
11. archive\_cvtran
12. archive\_dweldat
13. archive\_exadmn
14. archive\_excode
15. archive\_exdet
16. archive\_htagnt
17. archive\_htpar
18. archive\_land
19. archive\_legdat
20. archive\_lpmod
21. archive\_lpnbhd
22. archive\_oby
23. archive\_owndat
24. archive\_pardat
25. archive\_rcoby
26. archive\_sales
27. archive\_splcom
28. archive\_valclass

It’s important to note that the tables have datatypes for columns

1. **Errors to watch out for**
2. The Athena Tables and output parquet files are constricted by datatypes. So any parquet files which are uploaded in the future need to have the same column datatypes. If you want the datatypes to change then the Target files and athena tables need to be deleted and made fresh.
3. Because comfeat, comintext, comnt3 and htdates did not have files, they are not uploaded to proper athena tables yet. Since doing so might cause data type errors in the future since the parquet files uploaded will most likely have different datatypes
4. If the job is taking a long time, you can increase the workers up to the limits specified in the glue job configuration
5. **Paths**
6. Landing Zone path   
   s3://ccao-landing-us-east-1/iasworld/data/
7. Config file path

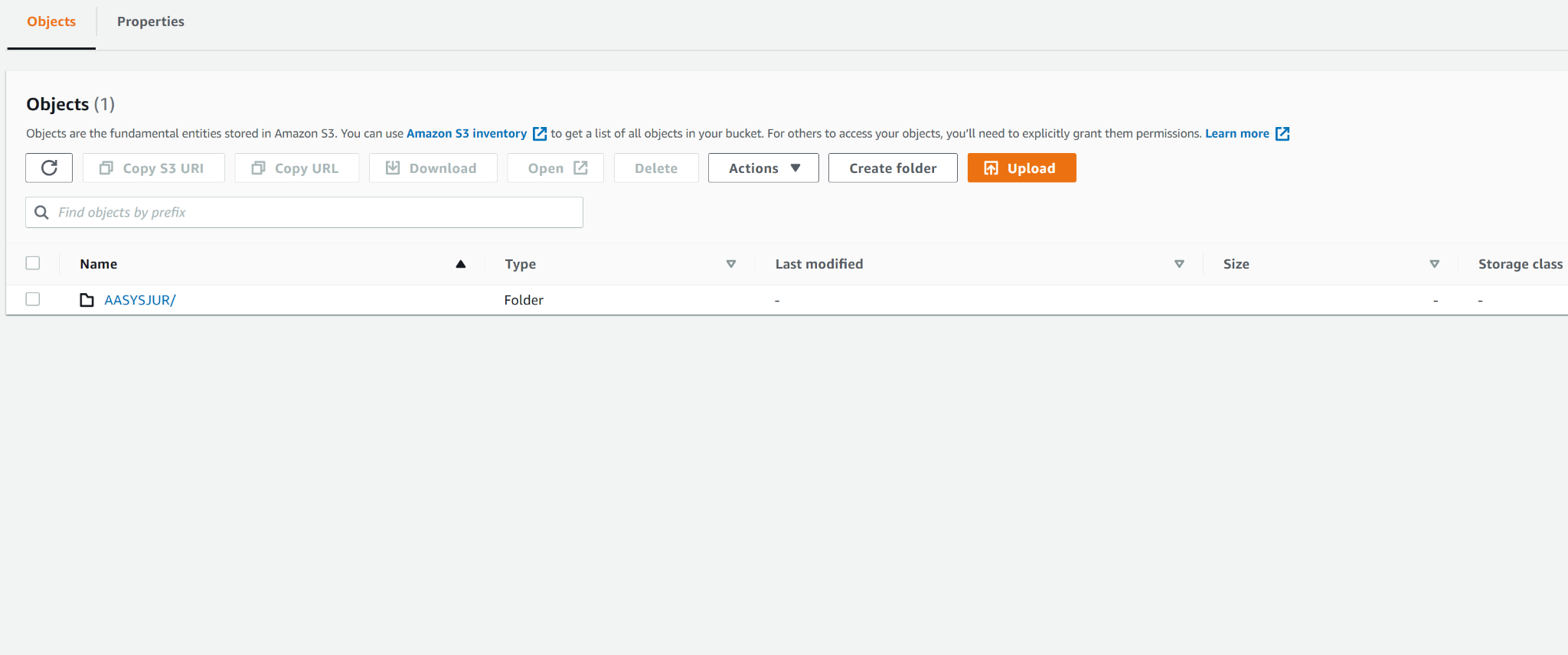
s3://ccao-staging-us-east-1/etl\_config.yaml

1. Target path  
   s3://ccao-staging-us-east-1/Target/
2. Auditlog path  
   s3://ccao-staging-us-east-1/AuditLogs/
3. Archive path   
   s3://ccao-staging-us-east-1/Archive/
4. Athena output folder  
   s3://ccao-staging-zone-athena-results-us-east-1
5. Glue Job Script Path  
   s3://ccao-glue-scripts-us-east-1
6. AWS Glue documentation for Spark compatibility  
   https://docs.aws.amazon.com/glue/latest/dg/release-notes.html

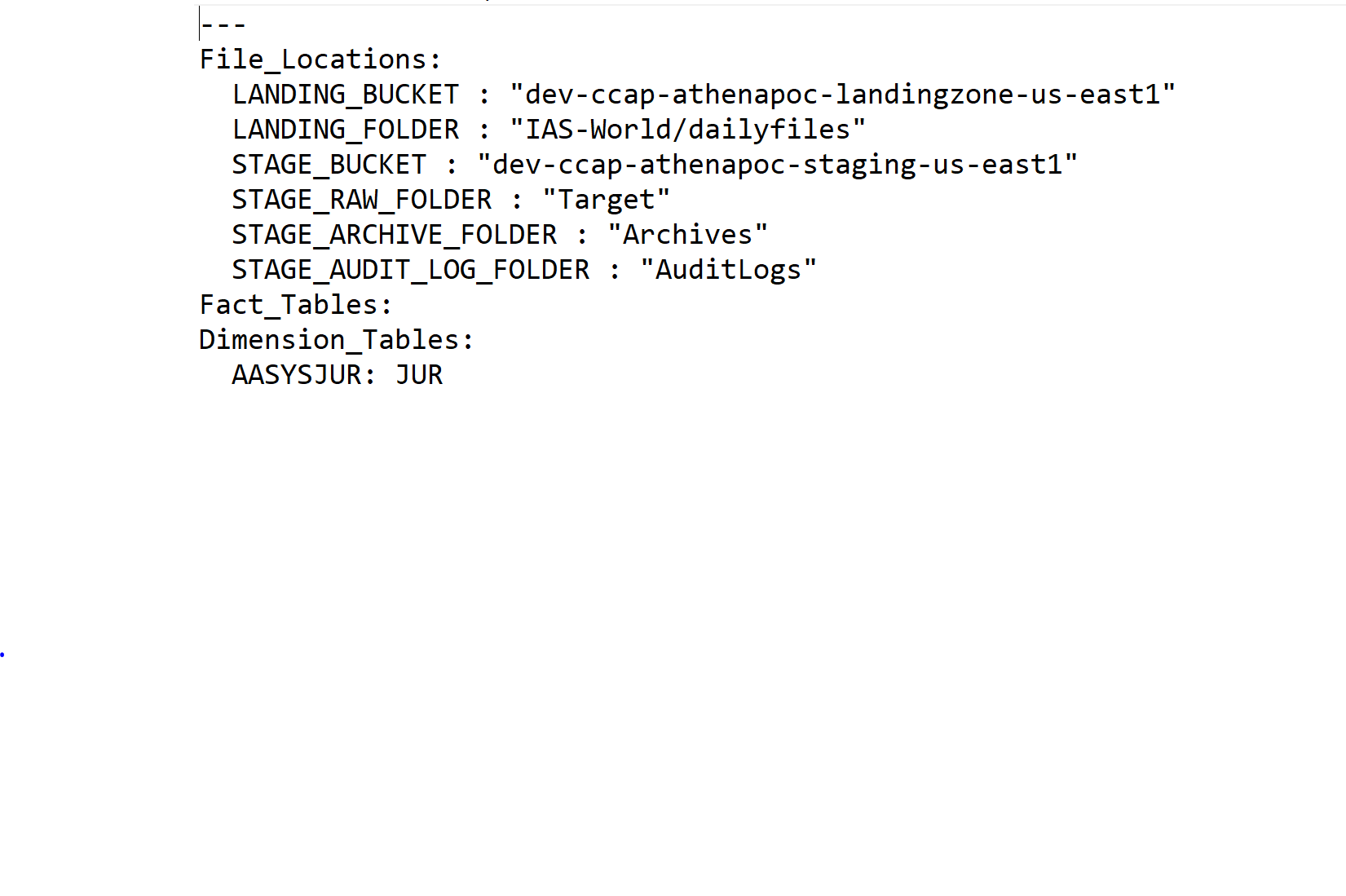
**8. Simple Walkthrough Example**

To setup a new file from scratch we will use an example of the table AASYSJUR

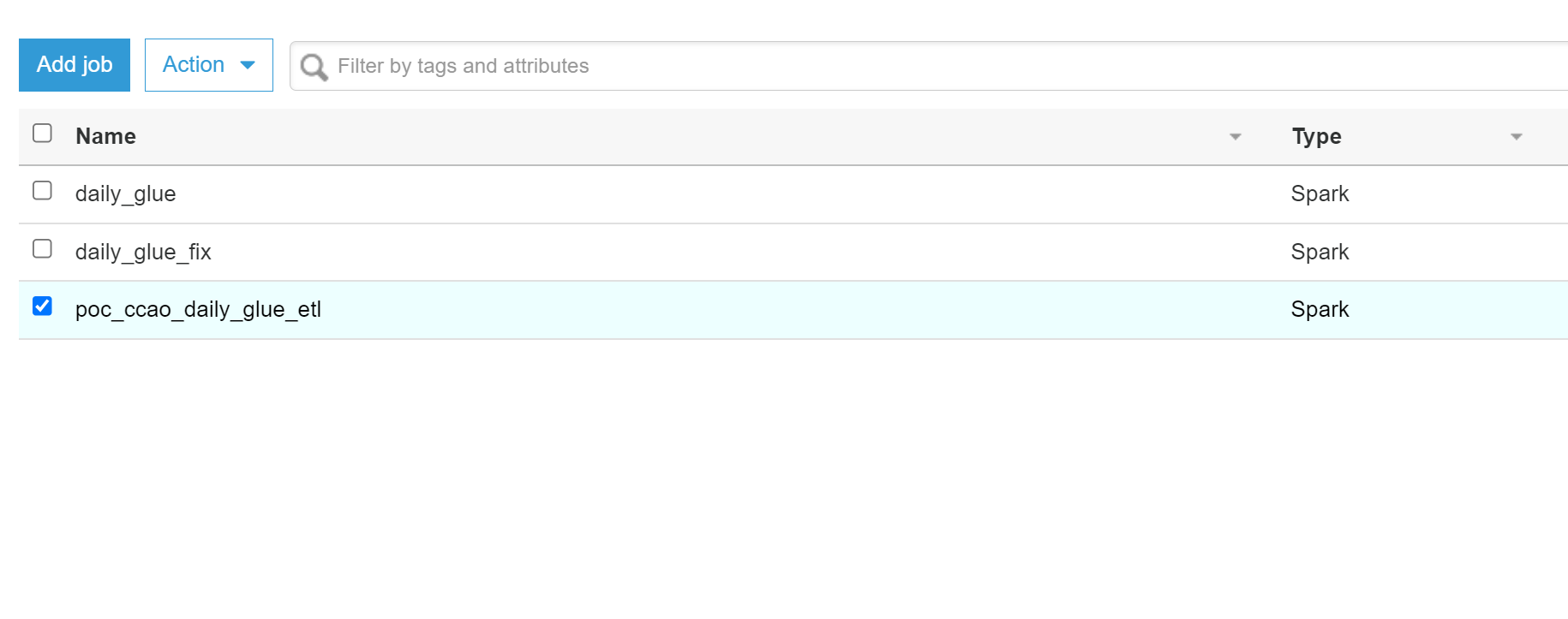
a) Upload the file to Landing Zone (ref 7a)



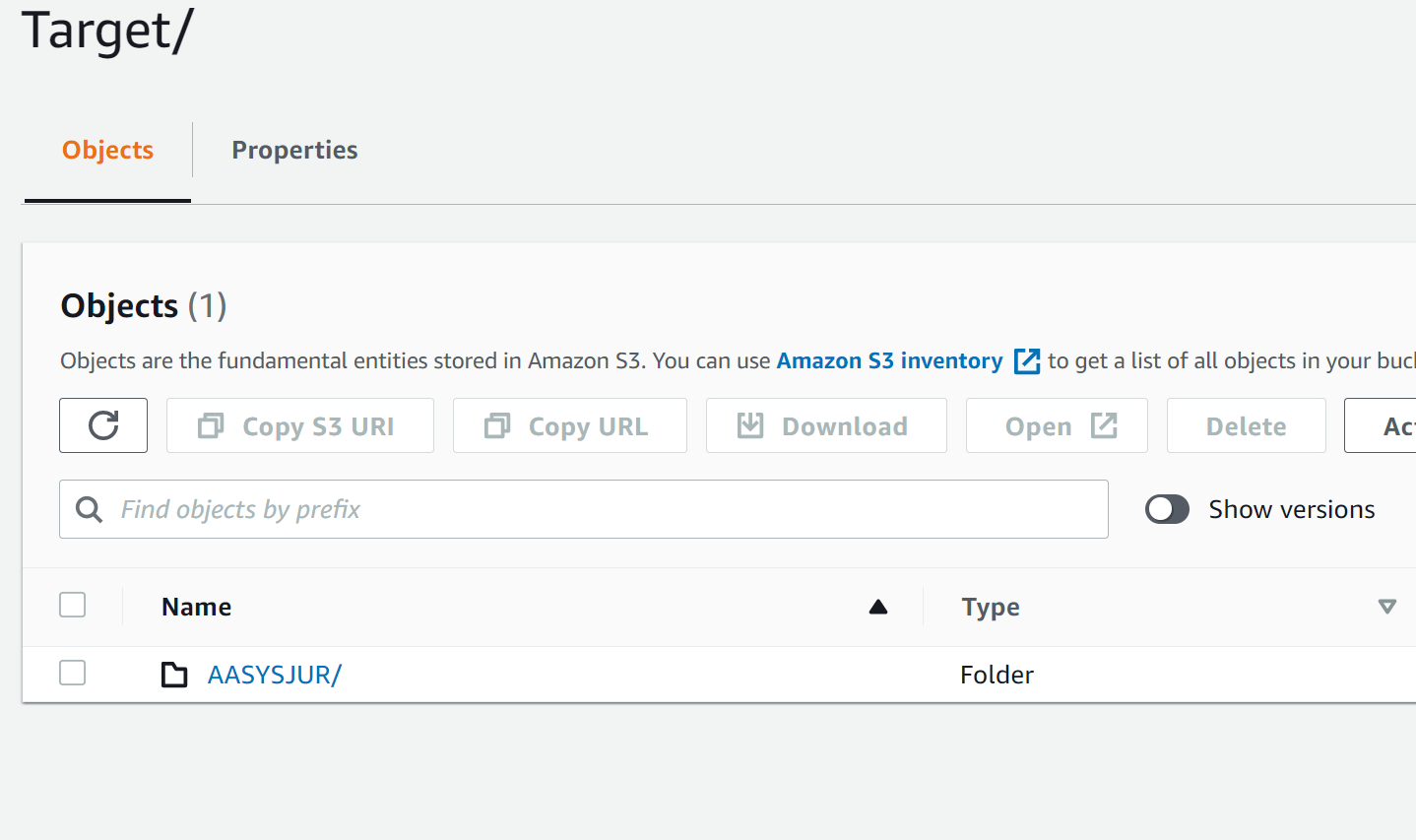
b) Upload config file which includes AASYSJUR and(or) any other files you wish to include(ref 3)

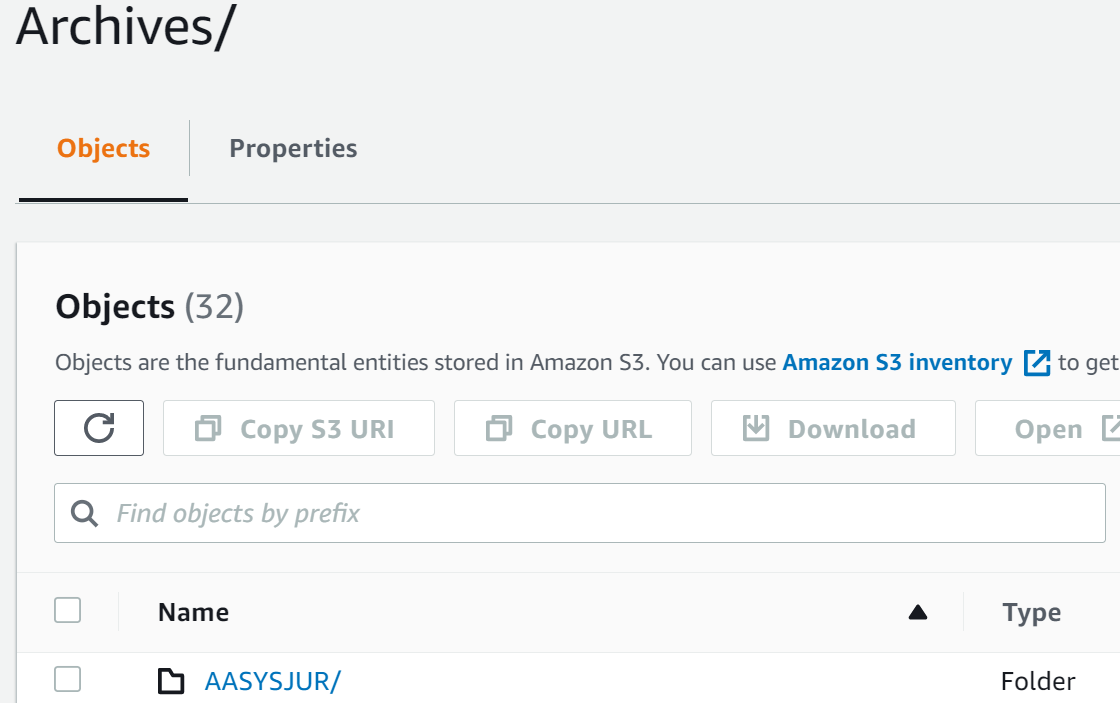


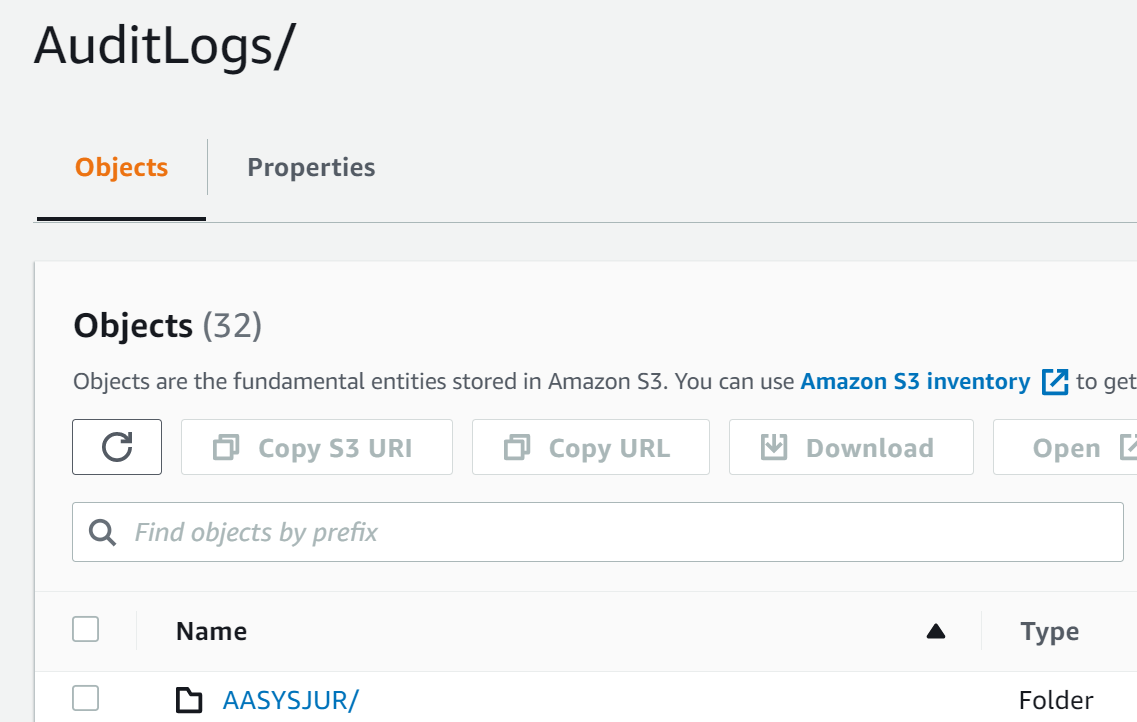
b) Run the Glue Job which is detailed in (ref 2), by left-clicking on poc\_ccao\_daily\_glue\_etl then left-clicking on action and next left-clicking on Run Job



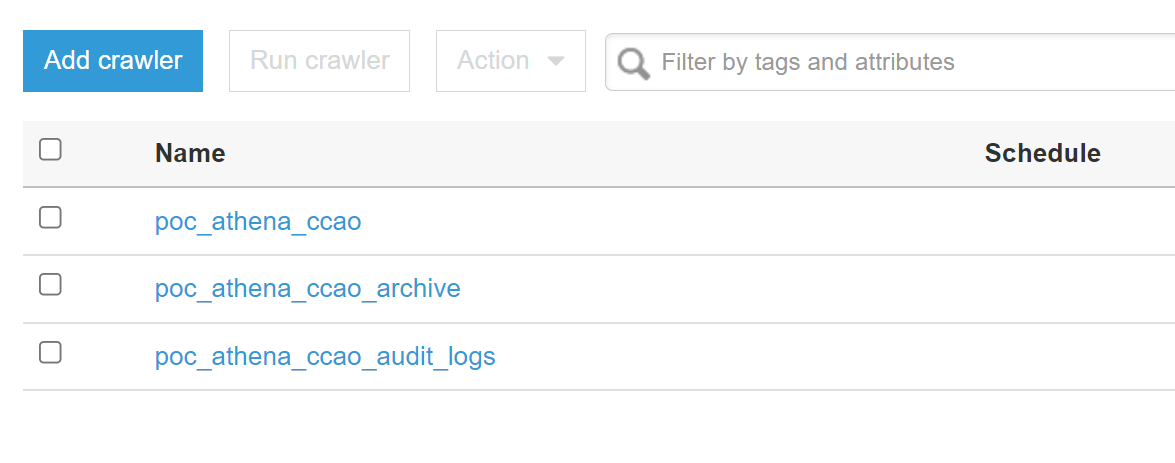
c) Upon successful completion of the job and output of the parquet file to theTarget(ref 7b),AuditLog(ref 7c) and Archive folders( ref 7d).







c) run the glue crawlers that creates Athena Tables for Target,AuditLog and Archive folders(ref 4)



Following which you can check to see if AASYSJUR was added to the athena table by looking in the Athena section of AWS, then left-clicking on aasysjur and selecting preview table. It should show one row.

